REMARKS

This Amendment and Response is intended to fully respond to the Final Office Action mailed August 11, 2005. In that Office Action, claims 1-4, 6-22, 24, 26, 27, 29-37, 39, 40, and 42-49 were examined, and all were rejected. More specifically, all claims are subject to a restriction/election requirement; claims 1-4, 6-22, 34-37, 39, and 40 stand rejected under 35 U.S.C. § 101 because the claimed invention is directed to non-statutory subject matter; claims 26, 27, and 29-33 stand rejected under 35 U.S.C. § 112, second paragraph, as being incomplete for omitting essential structural cooperative relationships of elements; claims 34, 35, and 36 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Burridge (USPN 6,918,106), hereinafter Burridge; claims 1-3, 6, 9, 11-20, 24, 26, 27, 29, 31, 37, 39, and 40 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Burridge in view of "Enterprise JavaBeans," published by Haefel, hereinafter Haefel; claims 4 and 10 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Burridge in view of Haefel as applied to claim 1, and further in view of "Method to Update Java Class Library in Client Computer at Runtime," published by IBM, hereinafter IBM; claims 7, 8, 30, 32, and 33 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Burridge in view of Haefel as applied to claims 1 and 26, and further in view of Chan (USPN 6,470,494), hereinafter Chan; and claims 21 and 22 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Burridge in view of Haefel as applied to claim 1, and further in view of Menachemi (U.S. Patent Application Publication 2002/0103810), hereinafter Menachemi. Reconsideration of these rejections, as they might apply to the original and amended claims in view of these remarks, is respectfully requested.

In this Response, claims 1, 19, 20, 22, and 26 have been amended; claims 34-37, 39, 40, and 42-49 have been canceled; and no new claims have been added. Claims 19, 20, and 22 have been amended for format and not for patentability reasons.

Election/Restriction Requirement

Applicant hereby elects Group I for prosecution pursuant to the election/restriction requirement. All claims related to Group II have been cancelled, but Applicants reserve the right to file the non-elected claims in a divisional, continuation, or other application.

Claim Rejections - 35 U.S.C. § 101

Claims 1-4, 6-22, 34-37, 39, and 40 stand rejected under 35 U.S.C. § 101 because the claimed invention is directed to non-statutory subject matter. Claim 1 has been amended to claim at least one tangible media. As such, amended claim 1 and all claims that depend from claim 1 are now allowable over the 35 U.S.C. § 101 rejections. Claims 34-37, 39, and 40 have been cancelled, and the 35 U.S.C. § 101 rejections related to these claims are now moot.

Claim Rejections – 35 U.S.C. § 112

Claims 26, 27, and 29-33 stand rejected under 35 U.S.C. § 112, second paragraph, as being incomplete for omitting essential structural cooperative relationships of elements. Claim 26 has been amended to include a customized library generator that relates the client composite list to the customized library. Thus, claims 26, 27, and 29-33 are now allowable over the 35 U.S.C. § 112 rejections.

Claim Rejections – 35 U.S.C. § 103

Claims 34, 35, and 36 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Burridge. Claims 34, 35, and 36 have been cancelled and the rejections pertaining to 34, 35, and 36 are now moot.

Claims 1-3, 6, 9, 11-20, 24, 26, 27, 29, 31, 37, 39, and 40 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Burridge in view of Haefel. Claims 37, 39, and 40 have been cancelled, and all rejections related to claims 37, 39, and 40 are now moot.

Applicants respectfully traverse the section 103 rejections for all other claims. The claims as amended preclude a finding of a prima facie case of obviousness because one or more of the requirements of a prima facie case is absent. Indeed, such a prima facie case can only be met when **all** of the following requirements are met: (1) the combined references must teach or suggest all the claim limitations; (2) there must be some suggestion or motivation in the references themselves (or in the knowledge available to those skilled in the art) to combine the references; and (3) there must be a reasonable expectation of success. See MPEP §§ 706.02(j) and 2143. In this case, neither Burridge nor Haefel teach or disclose 1) identifying one or more

client-needed types; 2) generating the customized library, including the one or more client-needed types, where the client-needed types comprise one or more of the application-referenced types that are not client-loaded types; or 3) separately sending the customized library and the application to the client.

The present invention includes a customized library management method and system for generating a customized library needed for execution of an application in a client system. In response to an identification of a given application, such as a request from the client system or an internal instruction of the server, the server determines the appropriate types to include in a library to be sent to the client based on certain parameters. The parameters may include, for example, the types referenced by the application; the types already "loaded" on the client system, and a device profile describing characteristics of the client system. The customized library includes a subset of the types, referred to in the exemplary embodiments as the client-needed types, that are required by the application; the client-needed types include all application-referenced types that are not already "loaded" on the client. A type is loaded on a client if the type is present, resides, or is stored at the client. As such, the client-needed types, in the customized library, do not include all application-referenced types, but only those types not already on the client.

Examiner relies heavily on Burridge. Burridge describes methods for optimizing a program during its runtime execution. See col. 2, 45-51 ("a method and apparatus for determining which program units are loaded *during execution* of a dynamically loaded program"). More particularly, Burridge creates a library file of program files "loaded" during execution of a main program unit. See col. 2, lines 56-64 ("creating at least one library file containing only application program files loaded *during the first execution*"). Burridge describes loading as loading a program unit at run time before the unit is used, similar to invoking an object before executing a contained method in the object. See col. 2, lines 38-39 ("Each reference program unit must be loaded at run time before the referenced unit is used").

To create the library, Burridge describes an offline loader that executes a main program unit. See col. 5, lines 16-18. The offline loader waits for the main program to request a program unit or file, similar to an object requesting another object, and loads the program unit from the

client source. See col. 5, lines 19-22 ("If the offline loader 26 requires a program unit that has not already been loaded, the offline loader 26 obtains the program unit from the pathnames specified.") The program unit already exists on the client or can be accessed by the main program already on the client. The offline loader then writes a copy of the program unit to a library. See col. 5, lines 22-25 ("the offline loader 26 writes a copy of the program file to an application library file"). The library file is then used for any subsequent executions of the main program. See col. 5, line 66 – col. 6, line 1 ("The library file created at reference numeral 38 is used as the input source for application program files in subsequent executions of the main program unit."). At a later execution of the main program, the runtime loader obtains the program units from the library file, which reduces the overhead in obtaining application programs from multiple sources. See col. 6, lines 5-9.

Haefel describes Java archives (JAR files). More particularly, Haefel describes deployment descriptors that can be used with JAR files to customize the behavior of software at runtime. See page 30, paragraphs 4-6.

Independent Claim 1, Claim 24, and Claim 26

Amended claims 1, 24, and 26 are allowable over Burridge and Haefel either in combination or alone. Neither Burridge nor Haefel teach or disclose 1) identifying one or more client-needed types; 2) generating the customized library, including the one or more client-needed types, where the client-needed types comprise one or more of the application-referenced types that are not client-loaded types; or 3) separately sending the customized library and the application to the client. Burridge, the primary prior art reference, and the present invention as defined in the claims are fundamentally different. The differences, and why the present application is allowable over Burridge, will become abundantly clear after the following discussion.

First, Burridge does not teach identifying one or more client-needed types. The present invention, as defined in the claims, comprises application-referenced types, client-loaded types, and client-needed types. Application-referenced types are types required by the application to execute. Client-loaded types are types that are already present, e.g., stored on, the client, and client-loaded types have yet to be invoked or used by the application. A client-needed type is an

application-referenced type that is not a client-loaded type, i.e., a type required by the application that is not already on the client. The present invention, as defined in the claims, identifies the client-needed types by determining which application-referenced types are not present on the client.

Burridge does not identify client-needed types. The library file described in Burridge, "contains all application program files 22 loaded during the execution of the main method." Col. 5, lines 26-28. To create the library file, the offline loader executes the main method and copies every file, as it is loaded, into the library file. See col. 5, lines 18-22. The offline loader simply loads every file needed by the main method into the library file. In essence, the offline loader stores, into the library, all "application-referenced" types. There is no mention in Burridge of identifying client-needed types, which are any application-referenced types not already on the client.

Applicants concede that Burridge teaches that, "if the program unit has not been found after the initial search at reference numeral 440, the runtime loader optionally searches an alternate source for the program unit." However, this process is not the same as identifying client-needed types. To determine client-needed types, Burridge must teach something akin to examining the application-referenced types and determining from any client-loaded types those application-referenced types not yet loaded on the client. Burridge simply does not include such disclosure.

Second, Burridge does not teach generating the customized library, including the one or more client-needed types, where the client-needed types comprise one or more of the application-referenced types that are not client-loaded types. As explained above, the library in Burridge includes every file needed by the main method. See col. 5, lines 26-28 ("contains all application program files 22 loaded during the execution of the main method."). The library in Burridge essentially contains all application-referenced types.

The customized library, of the present invention, is different because the customized library includes only client-needed types. The customized library, thus, has fewer than all the application-referenced types, as opposed to the library described in Burridge, because the customized library excludes any type already stored or present on the client.

Further, the library described in Burridge includes application types that are "loaded," applying the term "loaded" as explained in present invention. Burridge includes, in the library, types that are present on the client and used during the execution of the main program. See col. 4, lines 39-41 ("application program files required for execution of a dynamically loaded program are collocated in a library file.") and generally, col. 2, lines 20-25 (showing that if files are not found in local JAR files, then the a browser may search the web with a HTTP request). Therefore, the library taught in Burridge must include client-loaded types, which are explicitly excluded from the customized library claimed in the present invention.

Third, Burridge does not teach separately sending the customized library and the application to the client. The application, referred to as the main program unit or main method in Burridge, is already present on the client in Burridge. Burridge describes executing the main method that is to be optimized. See col. 5, lines 13-20. ("A user or developer determines which main method will be optimized...loads the main method and begins to execute it."). The main method must be present to execute the main method for optimization. Thus, the main method or application is never sent to a client that requested the application.

Further, the library in Burridge is also not sent to a client that requested the application. The library file is a collocation of files used by an application. See col. 11, lines 42-43 ("a novel method and apparatus for collocating dynamically loaded program files has been described."). The library file provides the system to decrease the memory footprint of the running application. See generally col. 2, lines 48-51. As such, the library file of Burridge is stored locally in the memory of the client executing the main method. The library file is never sent to a client.

In contrast, the present invention, as defined in the claims, sends both the application and the customized library to a client that requested the application. Burridge does not describe any communication between two computer systems involving the application and the library. Therefore, Burridge simple does not perform the same process as the present invention.

The present invention, as defined in the claims, and Burridge are not two interchangeable methods that do the same thing, but two separate methods that execute separate functions for separate purposes. For example, a user may request and application using the present invention, as defined in the claims. The computer process can determine what types are on the client. The

application and a customized library, with application-referenced types not on the client, can then be sent to the client requester. Once the application is stored at the client, a user could use the process described in Burridge to optimize the application. For example, the application could be executed. As types are loaded, either from the client-loaded types or the client-needed types in the customized library, the offline loader can place the types in another library. The application could then use the other library thereafter. As can be understood by this example, Burridge and the present invention, as defined in the claims, simply accomplish different things and are not comparable.

Haefel does not overcome the shortcomings of Burridge. Haefel describes deployment descriptors that can be used with JAR files to customize the behavior of software at runtime. See page 30, paragraphs 4-6. There is no mention in Haefel about 1) identifying one or more client-needed types; 2) generating the customized library, including the one or more client-needed types, where the client-needed types comprise one or more of the application-referenced types that are not client-loaded types; or 3) separately sending the customized library and the application to the client. Therefore, Haefel does not teach these elements of the claims.

While Haefel does describe using JAR files to shrink wrap third party components, there is no motivation to combine Haefel and Burridge. To the contrary, Burridge teaches away from the combination. Burridge desires to collocate program units in a single library. See col. 4, lines 40-42. Haefel describes packaging components for delivery to third parties. See page 90, paragraph 1. The combination of Burridge and Haefel would result in two libraries, a library or JAR file created for optimization and the JAR file received from a third party. The combination of the two libraries is contrary to the goal of Burridge, and thus, Burridge teaches away from this combination.

Further, Burridge does not teach sending the library or JAR file but keeping the library at the client to reduce memory allocation and optimize the function of the main method. See col. 2, lines 48-53. The combination of Burridge and Haefel suggests creating a JAR file using the method described in Burridge and sending the JAR file to another location. By sending the JAR file to another location, the purpose of Burridge is defeated, i.e., trying to optimize the execution

of an application at the client by collocating files at the client. For these reasons, there is no motivation to combine the references.

For the reasons stated above, independent claims 1, 24, and 26 are allowable over the prior art. Likewise, claims 2-4, 6-22, 27, 29-33 depend from the allowable independent claims and thus, are also allowable. Examiner is respectfully requested to remove the rejections to the amended claims and allow all claims now present.

Conclusion

This Amendment fully responds to the Final Office Action mailed on August 11, 2005. It is recognized that the Office Action may contain arguments and rejections that are not directly addressed by this Amendment due to the fact that they are rendered moot in light of the preceding arguments and amendments in favor of patentability. Hence, the failure, if any, of this Amendment to directly address an argument raised by the Examiner should not be interpreted as reflecting the Applicant's belief that such argument has merit. Furthermore, the claims of the present application may include other elements, not discussed in this Amendment, which are not shown, taught, or otherwise suggested by the art of record. Accordingly, the preceding arguments in favor of patentability are advanced without prejudice to other bases of patentability.

It is believed that no further fees are due with this Response. However, the Commissioner is hereby authorized to charge any deficiencies or credit any overpayment with respect to this patent application to deposit account number 13-2725.

In light of the above amendments and remarks, it is believed that the application is now in condition for allowance, and such action is respectfully requested. If the Examiner believes a telephone conference would advance the prosecution of this application, the Examiner is invited

to telephone the undersigned at the below-listed telephone number.

Respectfully submitted,

Date: November 14, 2005

Tadd F. Wilson
Reg. No. 54,544
MERCHANT & GOULD P.C.
P.O. Box 2903
Minneapolis, Minnesota 55402-0903
303.357.1651